REMARKS/ARGUMENTS

Claims 1-8, 10-17, 19-23, and 26-30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cohen (US Patent No. 5,751,614) in view of Deip. Claims 9, 18, 24, 25, 31, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cohen in view of Deip and further in view of Kabir (US Patent No. 6,538,657). Claims 1, 3, 6, 9, 10, 12, 15, 18, 19, 25, 26, and 32 have been amended in this Response to more clearly recite the present invention and improve readability. Claims 2, 4, 5, 7, 8, 11, 13, 14, 16, 17, 20-24, and 27-31 remain unchanged.

Applicants respectfully traverse the pending rejections based various combinations of Cohen, Diep, and Kabir, because each of these references fails to qualify as prior art with respect to the present invention. The present application claims priority back to the 8/16/95 filing date of U.S. Patent Application No. 08/516,036, which issued as U.S. Patent No. 5,742,840 (the '840 patent). This chain of priority also includes a continuation-in-part (CIP) application, U.S. Patent Application No. 09/382,402, which issued as U.S. Patent No. 6,295,599 (the '599 patent). As explained in the Applicants' response filed 11/15/06, none of the references Cohen, Diep, and Kabir qualifies as prior art in view of the 8/16/95 filing date of the present invention.[1]

The present invention is entitled to the 8/16/95 filing date for at least three important reasons. First, the '840 and '599 disclosures provide support for the claimed invention, contrary to the Examiner's assertions. Second, the support for the claimed invention found in the '840 and '599 disclosures clearly satisfies the requirements for written description and enablement under 35 U.S.C. § 112, paragraph 1. Third, the present invention is entitled to the 8/16/95 filing date despite the existence of a continuation-in-part application in the chain of priority. These reasons are explained in detail below.

---

[1] Cohen fails to qualify as prior art because the earliest filing date of Cohen associated with the feature cited by the Examiner is 2/29/96. The Examiner cites to the "mask" feature in Fig. 3 of Cohen. Cohen is a continuation-in-part (CIP) of parent application 08/444,814, filed 5/18/95. However, the "mask" feature cited by the Examiner was not disclosed in the parent application. That is, the "mask" feature appeared for the first time in the Cohen application filed 2/29/96, which is after the 8/16/95 priority date of the present application. As such, Cohen fails to qualify as prior art against the pending claims. Diep fails to qualify as prior art because Diep is a paper that appears to have been published in 1995. *See* Diep, copy right notice dated 1995. As such, Diep has not been shown to be published more than one year prior to the 8/16/95 priority date of the present invention and thus fails to qualify as prior art. Kabir also fails to qualify as prior art. Kabir is a continuation of parent application Ser. No. 09/289,783, filed

I. THE '840 AND '599 DISCLOSURES PROVIDE SUPPORT FOR THE CLAIMED INVENTION

Contrary to the Examiner's assertions, the '840 and '599 disclosures provide support for the claimed invention. The Examiner questions the 8/16/95 filing date of the present invention, alleging that no support for the claimed invention can be found in the '840 and '599 disclosures. The Examiner contends that no "single instruction" is taught to perform a masking or bitwise insertion operation. Specifically, the Examiner states:

> "However the masking or bitwise insertion operation is not taught and no means for a single instruction to perform this complex operation is taught by the '599 or '840 patents." Office Action dated 2/21/07, p. 8, paragraph 3.

The Examiner further contends that no "masking means" is taught for selectively determining whether one bit or bits is to be inserted or stored, and that without such "masking means," storage as taught by Applicants is merely conventional storage of data to memory. Specifically, the Examiner states:

> "Also the use of masking bit means is in neither patent teachings where the '599 and 840 patent merely sequentially store data in a register from a data path. There is no masking means to selectively separately determine one bit or bits is to be inserted while other means are used to determine whether the other bits of destination receive bit or bits. Therefore there is no support for the masking or bitwise insertion operation. Without any teachings for bitwise insert or masking means the storing is merely conventional storing such as shifting in data into a memory location." *Id.*

Applicants strongly disagree with the Examiner's contentions. The '840 and '599 disclosures clearly teach a single instruction that performs a masking or bitwise insertion operation. The '840 and '599 disclosures each includes an appendix filed as part of the original application. Both the '840 appendix and the '599 appendix describe embodiments of a "single instruction" that performs a masking or bitwise insertion operation. In fact, at least two such instructions are described: (1) "S.MUX.64.B.A.I" (Store multiplex octlet big-endian aligned immediate) and (2) "S.MUX.64.L.A.I" (Store multiples octlet little-endian aligned immediate). These two instructions are identified at pp. 150-157 of the '840 appendix (and pp. 123-125 and

---

4/9/99, which is a continuation of application. 08/563,059, filed 11/27/95. Thus, Kabir was filed after the 8/16/95 priority date of the present application and fails to qualify as prior art against the pending claims.

128-130 of the '599 appendix). Each of these instructions is an instance of a "single instruction" that performs a masking or bitwise insertion operation. Just as an example, a listing of the "S.MUX.64.B.A.I" instruction from p. 154 of the '840 appendix is reproduced below (highlighting added):

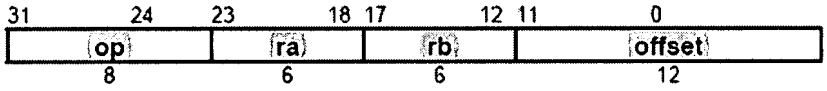
Operation codes

S.8.I[47]	Store byte immediate
S.16.B.A.I	Store double big-endian aligned immediate
S.16.B.I	Store double big-endian immediate
S.16.L.A.I	Store double little-endian aligned immediate
S.16.L.I	Store double little-endian immediate
S.32.B.A.I	Store quadlet big-endian aligned immediate
S.32.B.I	Store quadlet big-endian immediate
S.32.L.A.I	Store quadlet little-endian aligned immediate
S.32.L.I	Store quadlet little-endian immediate
S.64.B.A.I	Store octlet big-endian aligned immediate
S.64.B.I	Store octlet big-endian immediate
S.64.L.A.I	Store octlet little-endian aligned immediate
S.64.L.I	Store octlet little-endian immediate
S.128.B.A.I	Store hexlet big-endian aligned immediate
S.128.B.I	Store hexlet big-endian immediate
S.128.L.A.I	Store hexlet little-endian aligned immediate
S.128.L.I	Store hexlet little-endian immediate
S.AAS.64.B.A.I	Store add-and-swap octlet big-endian aligned immediate
S.AAS.64.L.A.I	Store add-and-swap octlet little-endian aligned immediate
S.CAS.64.B.A.I	Store compare-and-swap octlet big-endian aligned immediate
S.CAS.64.L.A.I	Store compare-and-swap octlet little-endian aligned immediate
S.MAS.64.B.A.I	Store multiplex-and-swap octlet big-endian aligned immediate
S.MAS.64.L.A.I	Store multiplex-and-swap octlet little-endian aligned immediate
S.MUX.64.B.A.I	Store multiplex octlet big-endian aligned immediate
S.MUX.64.L.A.I	Store multiplex octlet little-endian aligned immediate

The "S.MUX.64.B.A.I" instruction is clearly an example of a "single instruction" that performs a masking or bitwise insertion operation. The operation of the instruction is precisely defined in full detail, in a format that one of ordinary skill in the art would understand. *See* '840 appendix, pp. 154-157. *Also see* '599 appendix, pp. 128-130. Contrary to the Examiner's assertions, this definition provides a "masking means" to selectively determine which bits are to be stored. Specific portions of the definition as shown in the '840 appendix at pp. 154-157 and in the '599 appendix at pp. 128-130 are reproduced here (highlighting added) and described:

Format

S.size.order.align.I ra,rb,offset



•
•
•

Definition

```
def StoreImmediate(op,ra,rb,offset) as
    case op of
        S8I,
        S16LI, S16LAI, S16BI, S16BAI,
        S32LI, S32LAI, S32BI, S32BAI,
        S64LI, S64LAI, S64BI, S64BAI,
        S128LI, S128LAI, S128BI, S128BAI:
            function ← NONE
        SAAS64BAI, SAAS64LAI:
            function ← AAS
        SCAS64BAI, SCAS64LAI:
            function ← CAS
        SMAS64BAI, SMAS64LAI:
            function ← MAS
        SMUX64BAI, SMUX64LAI:
            function ← MUX
    endcase
```

As shown above, the "S.MUX.64.B.A.I" instruction performs a bit-level masking and write operation on a 64-bit portion of data. The inputs of the instruction include registers identified by "ra" and "rb," as well as an immediate value referred to as "offset" that is specified as a part of the instruction.

•
•
•

```
case op of
    S8I,
    S16LI, S16LAI, S16BI, S16BAI,
    S32LI, S32LAI, S32BI, S32BAI,
    S64LI, S64LAI, S64BI, S64BAI,
    SAAS64BAI, SAAS64LAI:
        rsize ← 64
    SCAS64BAI, SCAS64LAI, SMAS64BAI, SMAS64LAI, SMUX64BAI, SMUX64LAI:
        rsize ← 128
    S128LI, S128LAI, S128BI, S128BAI:
        rsize ← 128
endcase
```

•
•
•

```
a ← RegRead(ra, 64)
VirtAddr ← a + (offset11^50 || offset)
if align then
    if (VirtAddr and ((size/8)-1)) ≠ 0 then
        raise AccessDisallowedByVirtualAddress
    endif
endif
m ← RegRead(rb, rsize)
```

First, a series of initial steps sets the stage for the masked write operation. A 64-bit value is read from the register identified by "ra." The 64-bit value is combined with the immediate value "offset" to determine a memory location referred to as "VirtAddr." Then, a 128-bit value is read from a register pair identified by "rb." The 128-bit value is stored in the variable "m."

```
case function of
    NONE:
        StoreMemory(VirtAddr,size,order,m_size-1..0)
    AAS:
        b ← LoadMemory(VirtAddr,size,order)
        StoreMemory(VirtAddr,size,order,m_63..0+b)
        RegWrite(rb, 64, b)
    CAS:
        b ← LoadMemory(VirtAddr,size,order)
        if (b = m_63..0) then
            StoreMemory(VirtAddr,size,order,m_127..64)
        endif
        RegWrite(rb, 64, b)
    MAS:
        b ← LoadMemory(VirtAddr,size,order)
        n ← (m_127..64 & m_63..0) | (b & ~m_63..0)
        StoreMemory(VirtAddr,size,order,n)
        RegWrite(rb, 64, b)
    MUX:
        b ← LoadMemory(VirtAddr,size,order)
        n ← (m_127..64 & m_63..0) | (b & ~m_63..0)
        StoreMemory(VirtAddr,size,order,n)
endcase
enddef
```

Then, a "mask means" is used to perform a write operation to the specified memory location, by writing over certain bit fields identified by the mask means while leaving other bit fields unchanged. Here, the "mask means" is literally a mask that is stored at the lower 64 bits of the variable "m" ($m_{63...0}$). The mask identifies the bit fields that are to be written to the specified memory location "VirAddr." That is, for the bit fields having a predetermined value of "1" in the mask, corresponding bit fields of the data portion (which in this example is located in the upper 64 bits of the variable "m" ($m_{127...64}$)) are written to memory. Thus, the bit fields to be written to

memory are shown as "($m_{127...64}$ & $m_{63...0}$)."[2] Conversely, the inverse of the mask, expressed as (~$m_{63...0}$), identifies the bit fields that are to remain unchanged at the specified memory location "VirAddr." That is, for the bit fields that do not have a predetermined value of "1" in the mask, the unchanged data is retained by writing the original data back to the memory location. Thus, the bit fields to remain unchanged are shown as "(b & ~$m_{63...0}$)."[3] Finally, bit fields to be written (as identified by the mask) and the bit fields to remain unchanged are combined together, which is shown as "($m_{127...64}$ & $m_{63...0}$) | (b & ~$m_{63...0}$)."[4] The resulting combination is then written to the specified memory location "VirAddr."

As discussed in detail above, the "S.MUX.64.B.A.I" instruction is clearly an example of a "single instruction" that performs a masking or bitwise insertion operation. Another instruction – "S.MUX.64.L.A.I" (Store multiples octlet little-endian aligned immediate) – is an additional example of such a "single instruction" that performs a masking or bitwise insertion operation. Both the "S.MUX.64.B.A.I" instruction and the "S.MUX.64.L.A.I" instruction are fully described in the '840 and '599 disclosures.

To be sure, the operations performed by these instructions are not "merely conventional storing," as the Examiner alleges. *See* Office Action dated 2/21/07, p. 8, paragraph 3. Rather, each of these instructions is a true, bit-level masking instruction that has been unambiguously described in both the '840 and '599 disclosures. Accordingly, the '840 and '599 disclosures provide support for the claimed invention.

II. SUPPORT FOR THE CLAIMED INVENTION FOUND IN THE '840 AND '599 DISCLOSURES SATISFIES THE REQUIREMENTS FOR WRITTEN DESCRIPTION AND ENABLEMENT UNDER 35 U.S.C. § 112, PARAGRAPH 1

Furthermore, support for the claimed invention found in the '840 and '599 disclosures satisfies the requirements for written description and enablement under 35 U.S.C. § 112, paragraph 1. To satisfy the written description requirement, a patent disclosure must describe the

[2] The symbol "&" indicates a bitwise AND operation.
[3] The original data found at the memory location "VirAddr" has been read and stored in the variable "b." This is indicated by the operation "b ← LoadMemory (VirAddr,size,order)."
[4] The symbol "|" indicates a bitwise OR operation.

claimed invention in sufficient detail that one of ordinary skill in the art can reasonably conclude that the inventor had possession of the claimed invention. MPEP § 2163(I). As discussed previously, both the '840 and the '599 disclosure describe specific embodiments of a "single instruction" for performing a masking or bitwise insertion operation. For example, the "S.MUX.64.B.A.I" instruction is described in full detail, down to the bit level. This is a true masking instruction that has been unambiguously defined, as an illustrative embodiment of the invention. One of ordinary skill in the art would readily conclude, upon reviewing the '840 and '599 disclosures, that the Applicants had possession of the claimed invention.

To satisfy the enablement requirement, a patent disclosure when filed must contain sufficient information regarding the subject matter of the claims as to enable one of ordinary skill in the pertinent art to make and use the claimed invention. MPEP § 2164.01. Whether the enablement requirement is met depends on whether undue experimentation is necessary for one of skill in the art to practice the invention in light of the disclosure. *Id.* As discussed previously, the '840 and '599 disclosures describe specific embodiments of the claimed "single instruction" for performing a masking or bitwise insertion operation, including the "S.MUX.64.B.A.I" instruction. The '840 and '599 disclosures define this instruction in a manner that specifies the precise operation of the instruction, including exactly how every bit of data is obtained (*e.g.*, from specific registers, immediate values, memory locations etc.), operated on, and presented as output. One of ordinary skill in the art would not be required to perform undue experimentation – or any experimentation at all for that matter – to understand exactly how the "S.MUX.64.B.A.I" instruction is carried out as an illustrative embodiment of the invention. As such, the '840 and '599 disclosures clearly enable one of ordinary skill in the art to make and use the claimed invention.

Further evidence that the '840 and '599 disclosures satisfy both the written description and the enablement requirements is provided by way of a declaration from Mr. Korbin Van Dyke that is included with this Response. For example, Mr. Van Dyke states in his declaration at pp. 15-16, paragraphs 43 and 46, that:

"Therefore, I believe that each of the '599 patent, including the Zeus manual, and the '840 patent, including the Terpsichore manual, provide adequate written description and enablement as required by 35 USC § 112 for the elements of "an instruction path", "a data path", "an external interface operable to receive data from an external source and communicate the received data over the data path", "a register file operable to receive and store data from the data path and communicate the stored data to the data path", and "an execution unit coupled to the instruction and data paths and operable to decode and execute instructions received from the instruction path, wherein in response to decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register, the execution unit is operable to: (i) detect some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields; and (ii) cause the write-enabled data fields to be written to a specified memory location" of claim 1 (as amended) of the 10/757,516 patent application."

"During my evaluation of the media processor patent application, I have been impressed by the thoroughness and overall high-quality of the Zeus and Terpsichore manuals. The manuals provide clear and unambiguous descriptions of media processing systems and are thorough and well-written. The manuals provide comprehensive descriptions of instructions in complete architectural detail. The information in the manuals would have been readily understood and easily accessible to software engineers coding the media processing systems, and hardware engineers implementing microprocessors for use in the media processing systems, and that is exactly what architecture reference manuals should be. This is not surprising, since the '599 patent and 'the 840 patent each include an architecture manual that is intended to enable hardware engineers to do exactly that – design, build, and implement a media processor that would include circuitry for elements of "an instruction path", "a data path", "an external interface operable to receive data from an external source and communicate the received data over the data path", "a register file operable to receive and store data from the data path and communicate the stored data to the data path", and "an execution unit coupled to the instruction and data paths and operable to decode and execute instructions received from the instruction path, wherein in response to decoding a single instruction for writing data to memory based on a mask and data contained in at least one register, the mask comprising fields that each correspond to a field of the data contained in the at least one register, the execution unit is operable to: (i) detect some of the fields of the mask as having a predetermined value to identify corresponding fields of the data contained in the at least one register as write-enabled data fields; and (ii) cause the write-enabled data fields to be written to a specified memory location" as described in the Zeus and the Terpsichore architecture manuals."

III. THE PRESENT INVENTION IS ENTITLED TO THE ORIGINAL FILING DATE DESPITE THE EXISTENCE OF A CONTINUATION-IN-PART APPLICATION IN THE CHAIN OF PRIORITY

Finally, the present invention is entitled to the original 8/16/95 filing date despite the existence of a continuation-in-part (CIP) application in the chain of priority. As discussed previously, the chain of priority for the present application includes a CIP application that issued

as the '599 patent. In questioning the filing date of the present invention, the Examiner points to the fact that the '599 patent was filed as a CIP application. *See* Office Action dated 2/21/07, p. 8, paragraph 3. However, the mere existence of a CIP application in the chain of priority does not automatically cut off the benefit of an earlier filing date.

Under 35 U.S.C. § 120, a claim in a U.S. application is entitled to the benefit of the filing date of an earlier filed U.S. application if the subject matter of the claim is disclosed in the manner provided by 35 U.S.C. § 112, paragraph 1, in the earlier filed application. MPEP § 201.11(B), paragraph 2. This rule holds true for a CIP application. *See id.*, paragraph 5. It follows that the existence of a CIP application in a chain of priority does not necessarily cut off the benefit of an earlier filing date. As long as the claimed subject matter is disclosed in the CIP application, the CIP application can serve as a link in a chain of applications that entitles an application to the filing date of the earliest parent application in the chain.

Here, even though the application corresponding to the '599 patent was a CIP application, it disclosed the claimed invention in a manner that satisfies both the written description and enablement requirements of 35 U.S.C. § 112, paragraph 1.[5] This allows the CIP application to properly serve as a link in the chain of priority, entitling the present invention to the benefit of the 8/16/95 filing date. The fact that the '599 disclosure also included other subject matter, in addition to subject matter supporting the present invention, is simply irrelevant. In other words, it does not matter that the '599 disclosure may have also described other subject matter such as "transferring portion of register via a path that is narrower than the register used by the instruction execution means," as characterized by the Examiner. *See* Office Action dated 2/21/07, p. 6, last paragraph to p. 7, first paragraph. What matters with regard to the issue of the filing date of the present invention is whether the '599 disclosure included material that serves as proper support for a "single instruction" that performs a masking or bitwise insertion operation. As already discussed in detail in previous sections, such support is clearly found in the '599 disclosure, as well as the '840 disclosure.

[5] The sufficiency of the '840 and '599 disclosures to support the claimed invention is discussed in previous sections of this Response.

Accordingly, Applicants respectfully submit that the pending rejections based on various combinations of Cohen, Diep, and Kabir should be withdrawn. The present invention is entitled to the filing date of 8/16/95. Both the '840 and '599 disclosures provide proper support for the present invention to establish this filing date. Mr. Van Dyke's declaration is submitted in further evidence of the fact that the '840 and '599 disclosures describe the invention in sufficient detail to satisfy both the written description and the enablement requirement under 35 U.S.C. § 112, paragraph 1. Thus, the 8/16/95 filing date of the present invention is clearly established. Consequently, none of the reference Cohen, Diep, and Kabir qualifies as prior art against the present claims, and the pending rejections based on these references should be withdrawn.

CONCLUSION

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested. If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 202-756-8624.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account 500417 and please credit any excess fees to such deposit account.

Respectfully submitted,

McDERMOTT WILL & EMERY LLP

Michael A. Messina
Registration No. 33,424

Please recognize our Customer No. 20277 as our correspondence address.

600 13th Street, N.W.
Washington, DC 20005-3096
Phone: 202.756.8000 MAM:llg
Facsimile: 202.756.8087
Date: August 16, 2007
WDC99 1437936-1.043876.0155